Code Autocomplete Manual

Release 2.0.0

Jacques Lucke

July 02, 2016

Welcome to the official manual of the **Code Autocomplete** addon.

Main Project Page: https://github.com/JacquesLucke/code_autocomplete

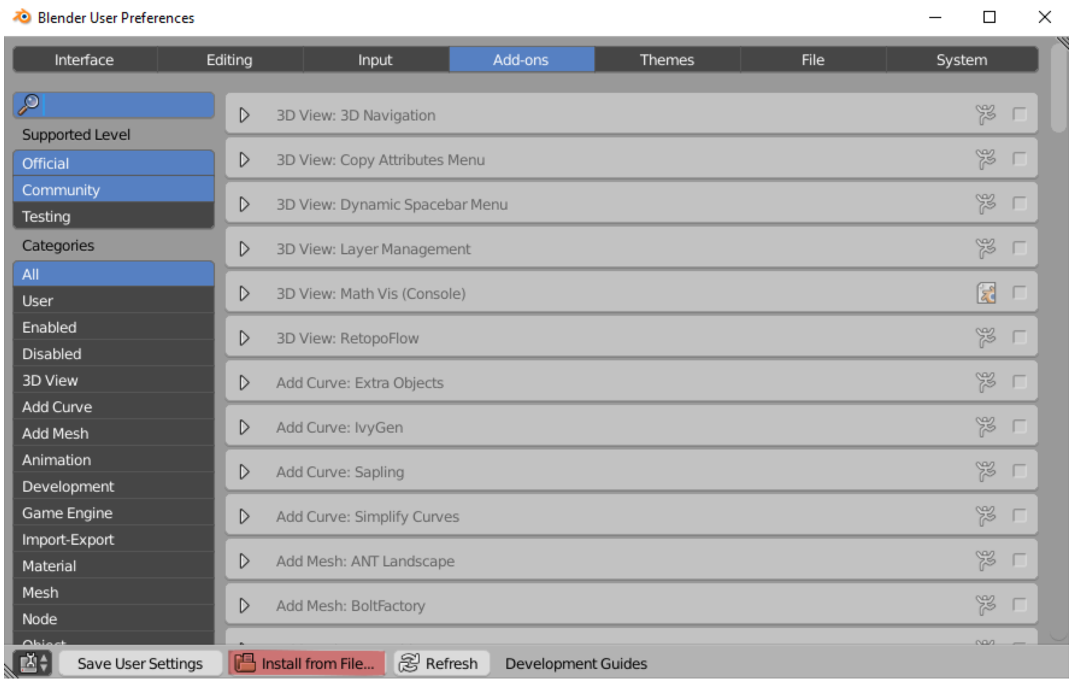
This addon uses the awesome Jedi autocompletion library that can be found here: http://jedi.jedidjah.ch/en/latest/

CHAPTER 1

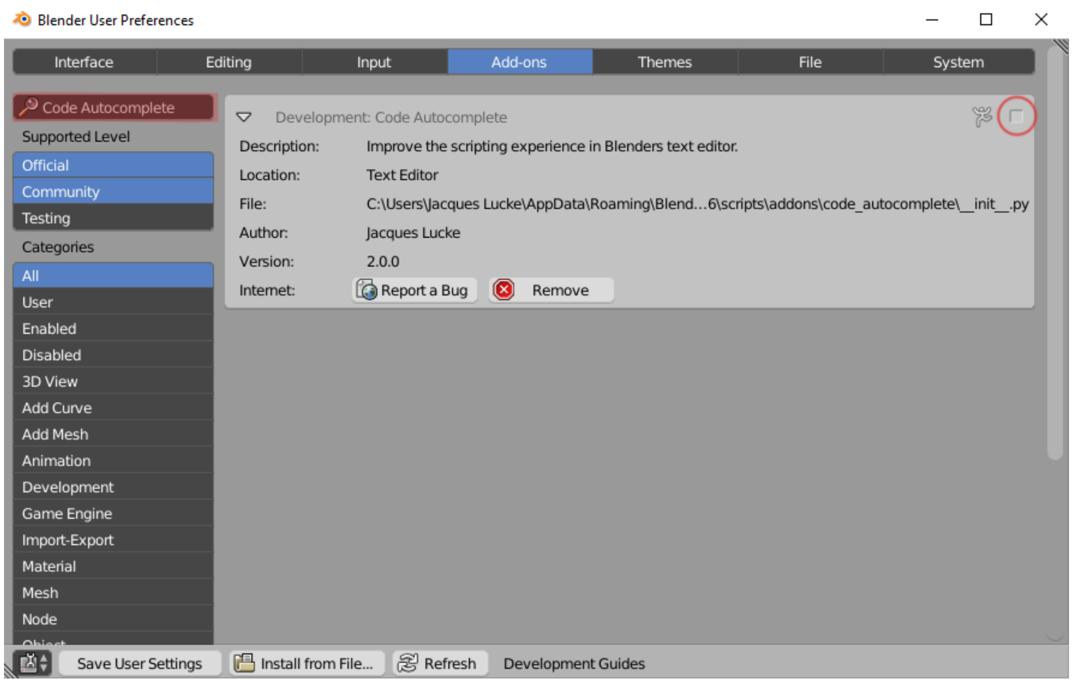
Setup

1.1 Installation

1. Download the .zip file (no need unzip it).
2. Open Blenders User Preferences and go to the **Add-ons** tab.
3. Click on **Install from File...** and select the .zip file you downloaded.



4. Enable the addon by clicking on the checkbox.



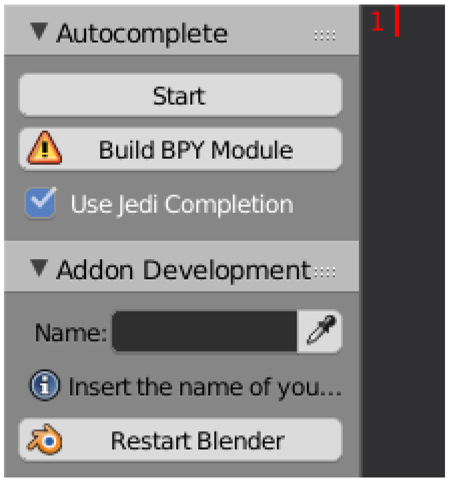
Development: Code Autocomplete

Description:	Improve the scripting experience in Blenders text editor.
Location:	Text Editor
File:	C:\Users\Jacques Lucke\AppData\Roaming\Blend...6\scripts\addons\code_autocomplete__init__.py
Author:	Jacques Lucke
Version:	2.0.0
Internet:	Report a Bug Remove

5. Click on **Save User Settings** so that the addon will be enabled when you start Blender the next time.

1.2 Check the Installation

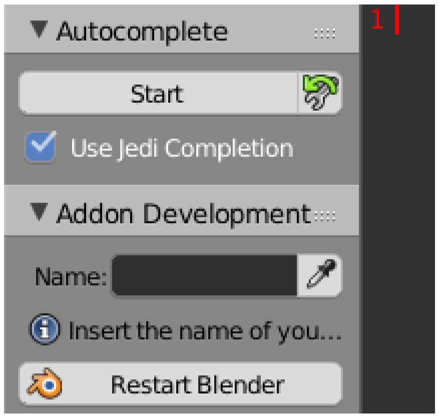
Open the text editor in Blender and look at the toolbar. You should see these two new panels:



When there is a error message *Jedi library not found* you have to use another .zip file in the beginning. Direct download from the github page is not supported, because the Jedi library is a submodule which isn't included in these downloads. You can either use an official release or use git to download the repository and its submodules (*git clone –recursive https://github.com/JacquesLucke/code_autocomplete.git*).

1.3 Finish the Setup

As last step you should click on the **Build BPY Module** button which creates a fake python module for large parts of the normal bpy module, so that Jedi can autocomplete types that are implemented in C instead of Python.



If there are any installation problems please report the issue on github: https://github.com/JacquesLucke/code_autocomplete/issues

CHAPTER 2

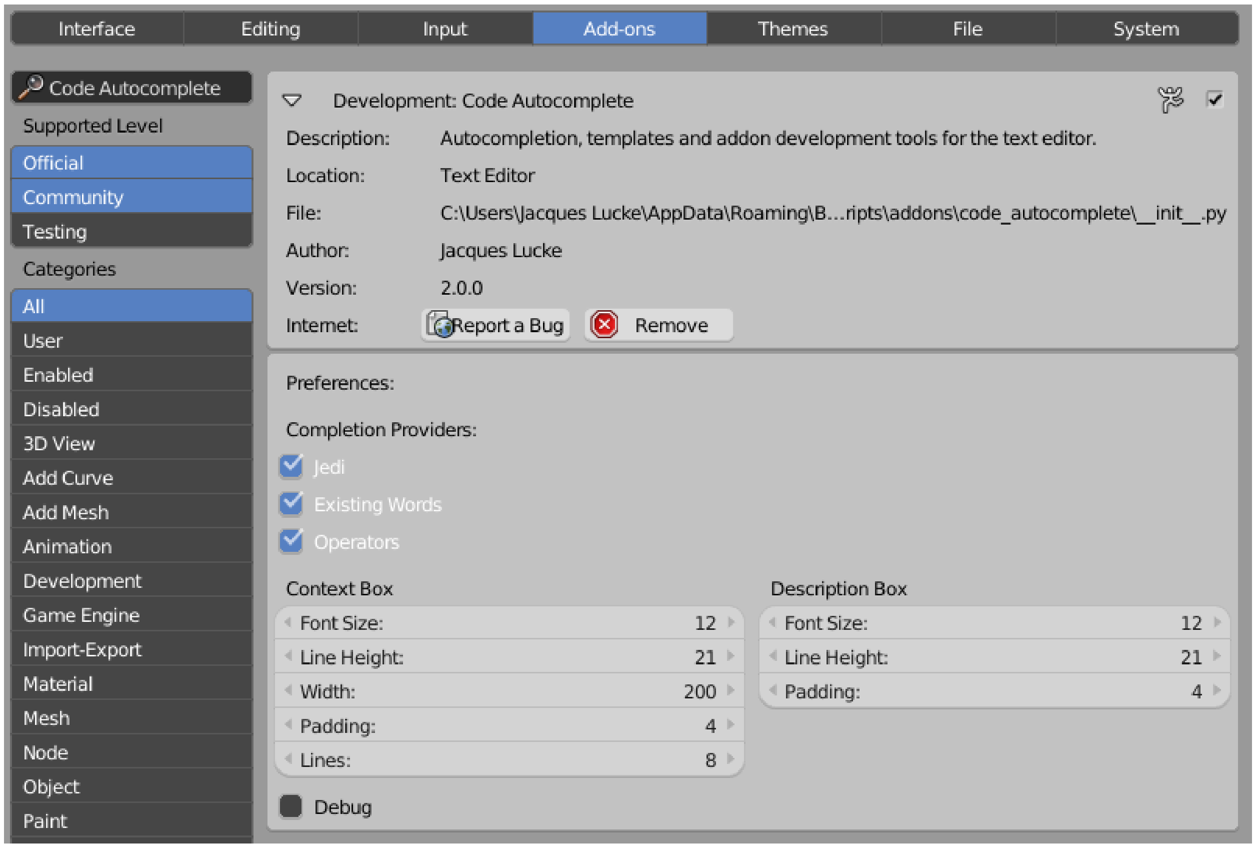
Preferences

There are a few settings you can change to customize the addon. The *Completion Providers* are functions that take your code and cursor information and try to find suggestions for you. Currently there three main completion providers:

1. **Jedi:** The Jedi autocompletion library analyses your python code and all the files you import to give you the best suggestions possible. Unfortunally, due to the dynamics of the python programming language this does not work in all cases.
2. **Existing Words:** This will collect all the words that are in the current text block and allows you to rewrite eg. names you previously wrote faster and with less spelling mistakes.
3. **Operators:** Whenever you want to call a blender operator using `bpy.ops.` this completion provider helps you and saves time because you don't have to look up the correct parameter names online in the api.

Beside these providers you get multiple properties to change the appearance of the **context and description box** in the text editor. The settings should be pretty self explanatory.

The **Debug** property is only for me if I want to activate some additional print statements in the addon which help me to track down bugs.



Development: Code Autocomplete

Description:	Autocompletion, templates and addon development tools for the text editor.
Location:	Text Editor
File:	C:\Users\Jacques Lucke\AppData\Roaming\B…ripts\addons\code_autocomplete__init__.py
Author:	Jacques Lucke
Version:	2.0.0
Internet:	Report a Bug Remove

Preferences:

Completion Providers:

- ✓ Jedi
- ✓ Existing Words
- ✓ Operators

Context Box

Font Size:	12
Line Height:	21
Width:	200
Padding:	4
Lines:	8

Description Box

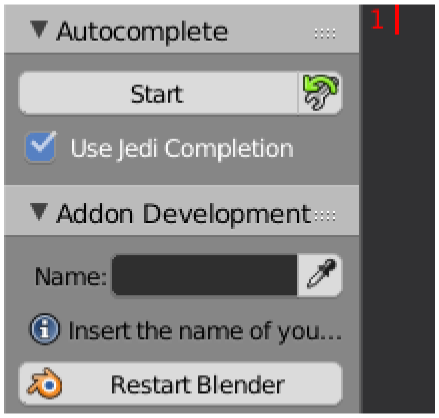
Font Size:	12
Line Height:	21
Padding:	4

Debug

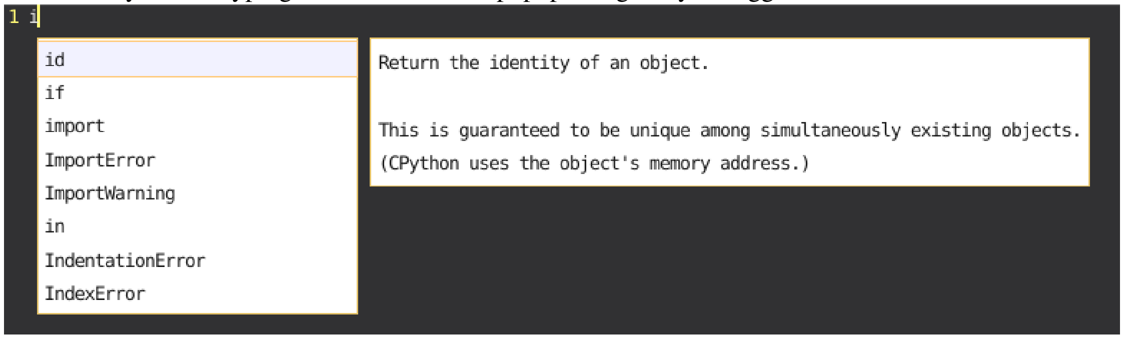
CHAPTER 3

Using the Autocompletion

To enable the autocompletion feature for the current session just click on the **Start** button.



As soon as you start typing a context box will popup that gives you suggestions.


```
i
```

```
id
if
import
ImportError
ImportWarning
in
IndentationError
IndexError
```

```
Return the identity of an object.

This is guaranteed to be unique among simultaneously existing objects.
(CPython uses the object's memory address.)
```

Normally the context box opens and closes automatically but you can also do this manually:

- **Open:** *shift ESC*
- **Close:** *ESC* or clicking somewhere else in the text editor

Navigation through the context box:

- *Arrow Up/Down:* Move the selected index in the arrow-direction.
- *Page Up/Down:* Same as arrow keys but skips a fixed a few items.
- *Home/End:* Jump to the first or last element.

- *Wheel Up/Down:* Move the selected index in the rotation direction. This only works when the mouse is on top of the context box.

Inserting a suggestion:

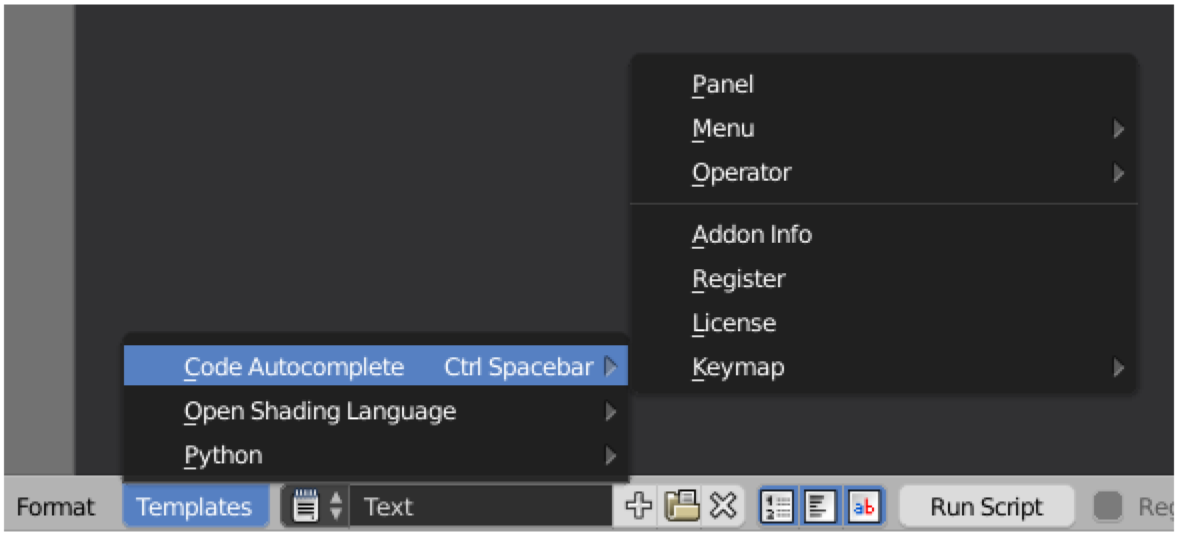
- Use *TAB/ENTER* to insert the currently selected suggestion.
- Left click on a suggestion will insert it too.

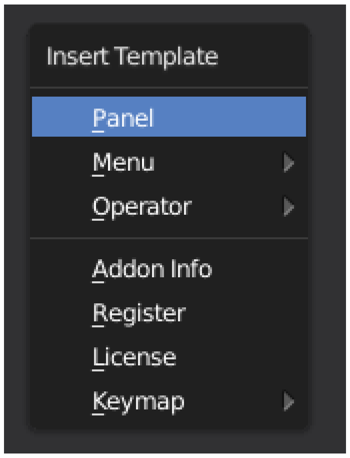
The Jedi library is not always fast. Particularly when it scans a new module it can take a second. After that it should be faster again. If you don't want these slow downs you can turn Jedi of by disabling the **Use Jedi Completion** property. Fortunally, the Jedi library is not the only *Completion Provider* in this addon. You will still be able to autocomplete words that already exist as well as the *bpy.ops* operators.

CHAPTER 4

Inserting Templates

The templates menu in the text editor contains a few more templates now. For faster access you can also use the *Ctrl Spacebar* shortcut.





Some templates have some extra settings you will be asked for. For example the name of a new Panel.

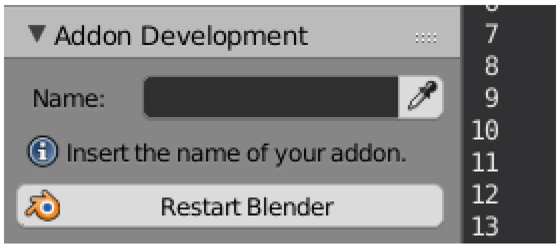
CHAPTER 5

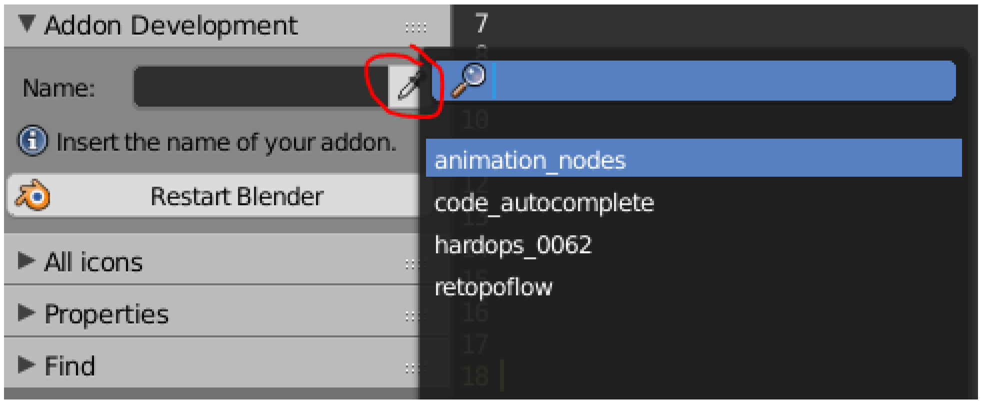
Addon Development

Code Autocomplete comes with several tools which simplify addon development inside of Blender a lot. Blender itself is not good for it, because it is hard to manage multiple files and to register everything correctly. This addon tries to overcome these difficulties by providing templates and operators to quickly create, extend and export code.

The Addon Development tools only work with addons which have their own folders (not single-file addons, these can be created without Code Autocomplete).

Before you can start coding you have to choose which addon you want to work on. You can either work on an existing one or create a completely new one. To select one that already exist you can write its name in the corresponding field or click on the eyedropper next to it to get a list of all installed multi-file addons.

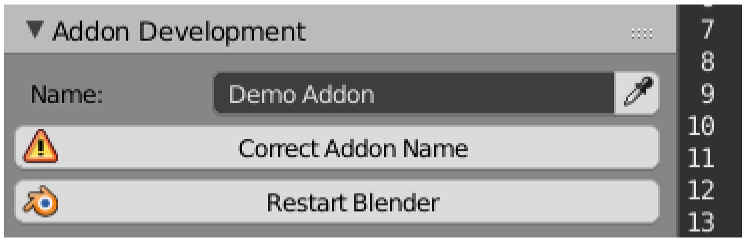


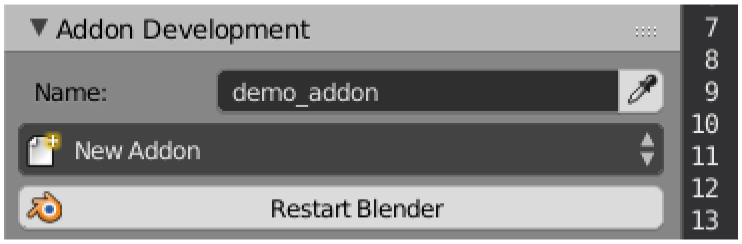


5.1 Creating a new Addon

To start a new addon you have to write the name it should have in the text box. Code Autocomplete will check automatically if the addon already exists and decides wether it should give you the options to create a new one or not. It will also not allow you to use addon names that may become problematic later.

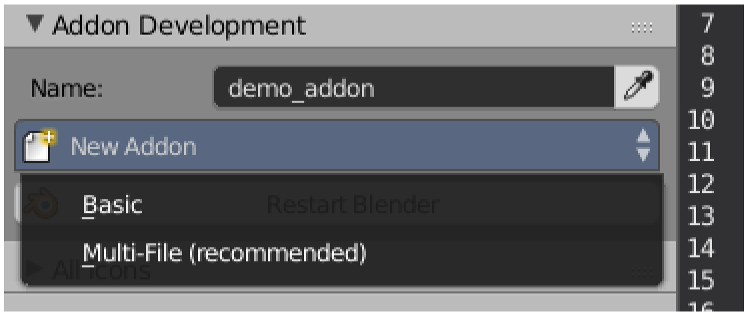
If you inserted a 'incorrect' addon name, a button to correct this name appears. Basically this operator replaces spaces with underscores and makes everything lower case (this is not the name the end user will see, and changing it later is easy).



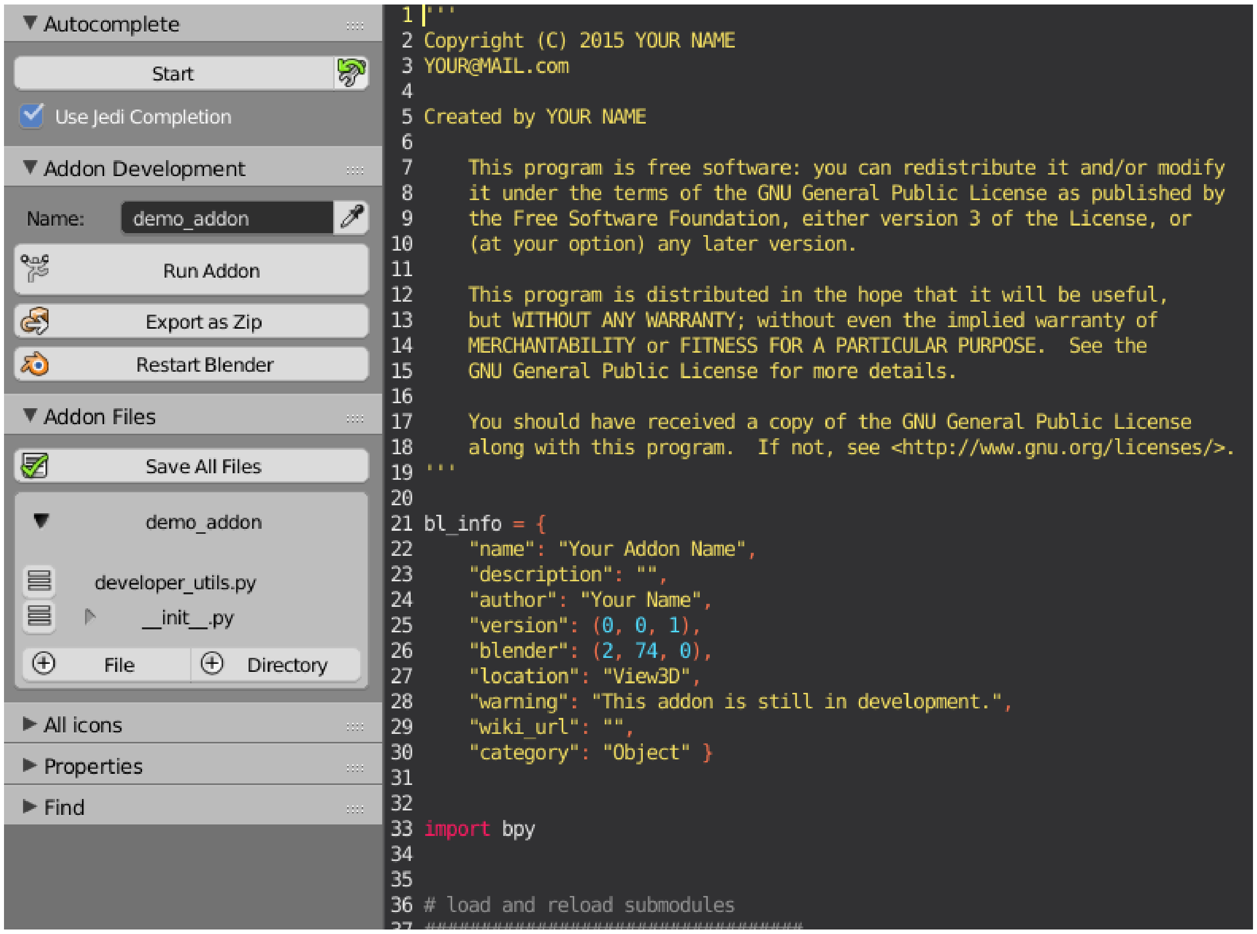


Once you have a correct name you get a new button to finally create the addon folder. You get two template options:

- **Basic:** This template contains only the minimum code that is needed to have a valid addon. There is the *bl_info* property as well as the *register* and *unregister* functions.
- **Multi-File:** This (recommended) addon template contains the same as *Basic*, but beside that there is a license and code that simplifies loading the different files of your addon. The loading code is the same that imports other addons like *Code Autocomplete* itself, but also the *Animation Nodes* addon.



As soon as you select a template the addon folder will be created. A new panel called *Addon Files* appears and the *__init__.py* file that is the heart of every addon is displayed in the text editor.



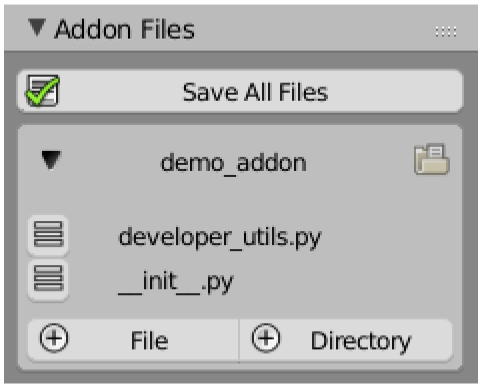
```
'''
Copyright (C) 2015 YOUR NAME
YOUR@MAIL.com

Created by YOUR NAME

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
'''

bl_info = {
    "name": "Your Addon Name",
    "description": "",
    "author": "Your Name",
    "version": (0, 0, 1),
    "blender": (2, 74, 0),
    "location": "View3D",
    "warning": "This addon is still in development.",
    "wiki_url": "",
    "category": "Object" }


import bpy


# load and reload submodules
```

5.2 Working on an Addon

5.2.1 Addon Files Panel

This panel gives you access to all the files and directories in your addon. You can create, open, rename and delete files. For more file manipulations you should open the addon folder in your default file browser using the folder icon.

The *Save All Files* operator looks which files you have currently opened and will save them. When you *Run* the addon this will happen as well.

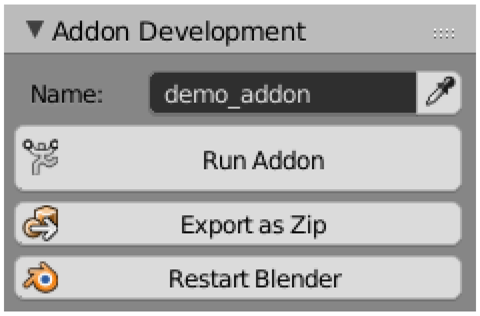


5.2.2 Addon Development Panel

When you click onthe *Run Addon* button all files will be saved first. Then your addon will be unregistered (if it has been registered before), reloaded and registered again. You should not use the normal *Run Script* button at the bottom.

From time to time it is useful to *Restart Blender* to have a clean working environment, so that you are not effected by earlier code executions. This operator currently only works on Windows.

When your addon is in a state so that you want to share it, the *Export as Zip* operator becomes very handy. After clicking you have to choose an output location and name. Now you complete addon will be packed in to a zip file that others can install.



CHAPTER 6

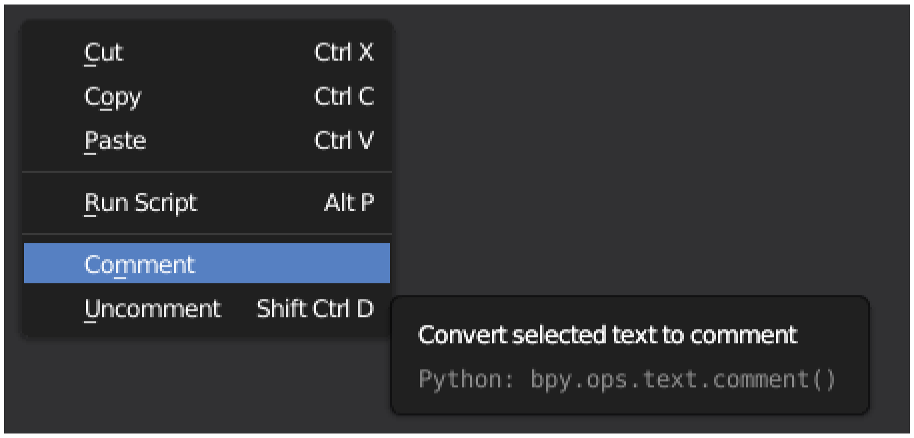
Misc

6.1 Select Whole String

When the cursor is inside of a string declaration and you press `ctrl Y` the whole string will be selected. This function is currently limited to the case when the string is only in one line.

6.2 Comment and Uncomment

I extended the right click menu in the text editor to enable you to comment the selected code lines out more quickly.

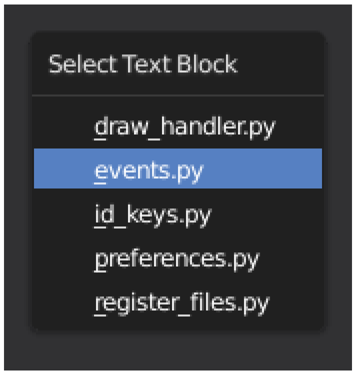


Convert selected text to comment

```
Python: bpy.ops.text.comment()
```

6.3 Switch Active Text Block

You can change the active text block with the `ctrl TAB` shortcut more quickly now. It opens a small menu with all the text blocks that are currently inside the file.



6.4 Correct Whitespace Inconsistency

This operator can be found in the *Format* menu. It does the same as *Convert to Spaces/to Tabs* but it chooses what to do automatically based on the *Tabs as Spaces* property in the *Properties* panel.